НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

«На правах рукопису»
УДК _____

«До захисту допущено»
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
“___” _____ 2019р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності _____
(код і назва)

на тему: Модифікація протоколів консенсусу для покращення масштабованості блокчейну.

Виконав (-ла): студент (-ка) 2 курсу, групи ФІ-83мп
(шифр групи)

Соколов Юрій Миколайович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. викладач каф. ММЗІ к.ф-м.н Фесенко А. В. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Фізико-технічний інститут
Кафедра математичних методів захисту інформації

Рівень вищої освіти: другий (магістерський) за освітньо–професійною програмою

Спеціальність: 113 «Прикладна математика»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
«___»_____201_ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

_____
(прізвище, ім'я, по батькові)

1. Тема дисертації _____
_____
_____,
науковий керівник дисертації _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від _____р. № _____

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження _____
_____

4. Предмет дослідження (Вхідні дані – для магістерської дисертації за освітньо–професійною програмою)
_____
_____
_____

5. Перелік завдань, які потрібно розробити _____
_____
_____
_____
_____

_____

6. Орієнтовний перелік ілюстративного матеріалу _____
_____
_____

7. Орієнтовний перелік публікацій _____
_____
_____

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____
(підпис) (ініціали, прізвище

* Консультантом не може бути зазначено наукового керівника магістерської дисертації.

РЕФЕРАТ

Кваліфікаційна робота містить: 54 стор., 8 рисунки, 0 таблиць, 37 джерел.

Метою даної кваліфікаційної роботи є модифікація протоколів консенсусу для покращення масштабованості блокчейну.

Завданням роботи є: дослідити протоколи консенсусу та проблеми масштабування протоколів консенсусу, модифікувати протоколи консенсусу для покращення масштабування.

Об’єктом дослідження є: інформаційні процеси в системах блокчейну.

Предметом дослідження є: властивість масштабування блокчейн протоколів консенсусу.

Результатом роботи є: Розподіл усіх вузлів на групи, використання довіреного середовища виконання, для зменшення кількості груп та учасників в кожній групі, використання передового алгоритму оптимізації обміну повідомленнями між комітетами.

Результат роботи може використовуватися при розробці нових блокчейн систем з можливістю масштабування.

МАСШТАБУВАННЯ, ПРОТОКОЛИ КОНСЕНСУСУ, БЛОКЧЕЙН, ШАРДИНГ, ДОВІРЕНЕ СЕРЕДОВИЩЕ.

РЕФЕРАТ

Квалификационная работа содержит 54 с., 8 рисунки, 0 таблиц, 37 источников.

Целью данной квалификационной работы является модификация протоколов консенсуса для улучшения масштабируемости блокчейну.

Задачей работы является: исследовать протоколы консенсуса и проблемы масштабирования протоколов консенсуса, модифицировать протоколы консенсуса для улучшения масштабирования.

Объектом исследования являются: информационные процессы в системе блокчейн.

Предметом исследования: проблемы масштабирования блокчейну.

Результатом работы являются: Распределение всех узлов на группы, использование доверенной среды выполнения, для уменьшения количества грепп и участников в каждой группе, использование передового алгоритма оптимизации обмена сообщениями между комитетами.

Результат работы может использоваться при разработке новых блокчейн систем с возможностью масштабирования.

МАСШТАБИРУЕМОСТЬ, ПРОТОКОЛЫ КОНСЕНСУСУ, БЛОКЧЕЙН, ШАРДИНГ, ДОВЕРЕННАЯ СРЕДА.

ABSTRACT

Qualifying work contains: 54 pages, 8 figures, 0 tables, 37 sources.

The purpose of this qualification is to modify consensus protocols to improve blockchain scalability.

The job is to: explore consensus protocols and the problems of scaling consensus protocols, modify consensus protocols to improve scaling.

The object of the study is: scaling consensus protocol blockchain property.

The result is: Division of all nodes into committees, use of trusted execution environment, to reduce the number of committees and participants in each committee, use of advanced algorithm for optimizing messaging between shards.

The result of the work can be used in the development of new blockchain systems with scalability.

SCALABILITY, CONSENSUS PROTOCOLS, BLOCKCHAIN, SHARDING, TRUSTED ENVIRONMENT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

PBFT - Practical Byzantine Fault Tolerance (укр. Практичний протокол візантійської угоди)

BFT - Byzantine Fault Tolerance (укр. Візантійська відмова)

TEE – Trusted Execution Environment (укр. Довірене середовище виконання)

$\binom{n}{c}$ – функція знаходження комбінацій

AHL - Attested HyperLedger (укр. Атестований HyperLedger)

ВСТУП

Актуальність дослідження. Актуальність даного дослідження полягає у тому, що протоколи які базуються на вирішенні візантійських відмов не спроможні масштабування деяку, відносно невелику, кількість вузлів причиною цього є надлишковий обмін повідомленнями між вузлами. Надлишковий обмін повідомленнями між вузлами спричиняє до нелінійного зменшення кількості транзакцій з часом, а використовуючи методи запропоновані в даній роботі, можна досягти масштабованості даних протоколів та сталу швидкість транзакцій.

Метою дослідження є модифікація протоколів консенсусу для покращення масштабованості блокчейну. Для досягання мети потрібно було виконати такі завдання:

1) провести огляд опублікованих джерел за тематикою дослідження;
2) провести дослідження протоколів консенсусу з вирішеними проблемами масштабування
3) запропонувати методи вирішення проблеми масштабування.

Об'єктом дослідження є: інформаційні процеси в системах блокчейну.

Предметом дослідження є: властивість масштабування блокчейн протоколів консенсусу.

Практичне значення результатів полягає в тому, що використовуючи методи запропоновані в даній роботі можна створювати системи блокчейну які будуть захищені, масштабуватися і не втрачати швидкості обміну транзакціями.

1 ЗАГАЛЬНА ІНФОРМАЦІЯ ПРО МАСШТАБОВАНІСТЬ БЛОКЧЕЙНУ

1.1 Загальні відомості про блокчейн

Згідно з загальним визначенням блокчейн (англ. Blockchain), або ланцюжок блоків транзакцій – це розподілена база даних, що зберігає впорядкований ланцюжок записів (так званих блоків), що постійно довшає. Дані захищено від підробки та спотворення. Кожен блок містить часову позначку, геш попереднього блока та дані транзакцій, подані як геш-дерево.

Учасники мереж блокчейну використовують консенсус через однорангову мережу (англ. Peer-to-peer) для узгодження кожного блоку транзакцій. Кожен блок містить упорядкований набір транзакцій та посилання (геш) на попередній блок ланцюга. Відхід гешей до попередніх блоків дозволяє нам просуватися через історію транзакцій. Незмінюваність блоків, «похованих» у блокчейні, ймовірно гарантована, оскільки модифікація блокує недійсні геші всіх нових блоків у ланцюзі.

Транзакція є найменшою можливою зміною (атомарною зміною) стану системи.

Нові блоки додаються до системи відповідно визначеному набору правил, встановлених протоколом. Важливою функцією цих правил є захист від атак зловмисників на блокчейн і досягнення консенсусу у випадку появи декількох варіантів ланцюга блоків.

1.2 Досягання консенсусу в блокчейн системах

В потенційно шкідливих, візантійських середовищах, часто використовуються додаткові методи забезпечення цілісності, та безпеки.

Тому, що окремі вузли можуть вийти з ладу, зловмисно поводитися, діяти проти спільної мети, або мережевий зв'язок може перерватися. Таким чином, для надання безперервної послуги вузли запускають протокол консенсусу, стійкий до відмов, щоб гарантувати, що всі вони узгоджують порядок внесення записів до блокчейн. Оскільки весь блокчейн виконує функції надійної системи, він повинен бути надійним, стійким та захищеним, забезпечуючи такі властивості, як доступність, надійність, безпека, конфіденційність, цілісність тощо [1].

Протокол блокчейну забезпечує це шляхом реплікації даних та операцій з використанням багатьох вузлів. Реплікація може мати багато ролей, але блокчейн тиражує дані лише для стійкості, а не для масштабування. Усі вузли, в принципі, підтверджують інформацію, яку потрібно додати до блокчейн; ця функція стимулює довіру всіх вузлів до того, що блокчейн в цілому працює правильно.

Оцінюючи протокол blockchain, важливо чітко визначити основні припущення щодо довіри або модель безпеки. Це визначає середовище, для якого розроблений протокол, і в якому він відповідає своїм гарантіям. Такі припущення повинні охоплювати всі елементи системи, включаючи мережу, наявність синхронізованих годин та очікувану (неправильну) поведінку вузлів.

Для кращого розуміння причин помилок необхідно визначити, яким чином процеси та комунікаційні канали зв'язку, що складають систему, можуть фактично некоректними за певною моделі системи.

Для дослідження задачі консенсусу розподілену систему зазвичай представляють як статичну, обмежену кількість процесів $\Pi = \{p1, p2, .., pn\}$, де звя'зок між процесами відбувається шляхом передачі повідомлень за допомогою надійного каналу точка-точка (англ. point-to-point). Процес є абстрактною частиною розподіленої системи, яка здатна виконувати обчислення [2]. Термін коректно використовується, лише якщо впродовж усього терміну виконання конкретної компоненти, такої як процес або

зв’язок між процесами, не буде жодної помилкової поведінки або відхилення від встановлених правил протоколу.

Для кращого розуміння причин помилок необхідно визначити, яким чином процеси та комунікаційні канали зв’язку, що складають систему, можуть фактично некоректними за певною моделі системи. Назвемо протокол f**–стійким**, якщо він допускає не більше ніж f некоректних процесів з усіх n процесів системи. Наступний список є певним узагальненням типів збоїв або помилок (але неповним), які можуть виникати під час функціонування системи:

– **Повний вихід з ладу** (англ. crash failure). Найпростіша модель помилки, за якої компонента системи зазнає збою і вже ніколи не відновиться.

– **Відмова виконання** (англ. omission failure). За такої моделі некоректні компоненти можуть не виконувати дії, такі як надсилання повідомлень або виконання обчислень. Компоненти зі здатністю відновлення після помилки також потрапляють до цієї категорії.

– **Невиконання термінів** (англ. timing failure). Помилки невиконання термінів виникають при порушенні припущень щодо синхронізації. В асинхронній системі цей тип помилок не має значення.

– **Візантійська відмова** (англ. Byzantine failure). Візантійська відмова (яку іноді також називають довільною помилкою) дозволяє компоненту довільно відхилятись від очікуваної поведінки, 2 а отже, і робити зловмисні дії. Ця категорія включає копіювання або модифікацію вмісту повідомлень, надсилання небажаних повідомлень та тимчасову або постійну імітацію будь-якого типу із зазначених раніше відмов.

Тиржування стану машини. Завдання досягнення та збереження консенсусу між розподіленими вузлами можна описати двома елементами:

1. Тиражування стану машини, де зазвичай потрібно досягти консенсусу як для вхідних даних, так і для їх порядку множиною детермінованих машин з тиражованим станом, таким чином, щоб всі

копії отримували та обробляли абсолютно однакову послідовність запитів.

2. Консенсус-протокол для розповсюдження запитів серед вузлів таким чином, що кожен вузол виконує однакову послідовність запитів у своєму екземплярі служби.

У літературі «консенсус» означає традиційно лише завдання досягнення згоди щодо одного єдиного запиту (тобто першого), тоді як **«атомна трансляція»**[4], це такий порядок трансляції, де повідомлення, надіслані до множини процесів, повинні доставлятися процесами в тому ж самому загальному порядку, передбачає узгодження послідовності запитів, необхідних для тиражування стану машини. Але оскільки між ними існує тісний зв'язок (послідовність випадків консенсусу забезпечує атомну трансляцію), термін "консенсус" частіше насправді означає атомну трансляцію, особливо в умовах блокчейн. Тут ми приймаємо цю термінологію, а також використовуємо "транзакцію" та "запит" як синоніми для одного з повідомлень, що доставляються в атомній трансляції або іншими словами в загальному порядку трансляції.

1.3 Задача візантійських генералів

Задача візантійських генералів — одна із задач криптографії, а саме: взаємодія декількох віддалених абонентів, які отримали накази з єдиного центру. Частина абонентів, разом із центром, можуть бути ворогами. Необхідно виробити єдину стратегію дій, переможну для лояльних абонентів. Протокол розв'язання цієї задачі називають протоколом **візантійської угоди** або **протоколом візантійських генералів**.

Ідея візантійського консенсусу з'явилася в 80-х роках минулого століття. Його суть полягає в наступному. Візантія напередодні битви. Армія

візантійців складається, наприклад, з 4 легіонів, які знаходяться один від одного на відстані. У певний час кожен з генералів легіонів отримує від керівного центру наказ йти в атаку або відступати.

Розвиток подій наступний:

1. Якщо усі легіони атакують - вони перемагають;

2. Якщо усі легіони відступають - вони зберігають людей (теж вдалий результат);

3. Якщо частина атакує, частина відступає - армія зазнає поразки.

Завдання зрозуміле, але де гарантія, що серед генералів немає зрадників, які виконають наказ навпаки? І де гарантія, що сам головнокомандувач не опиниться зрадником, відправивши різні накази різним генералам? Висновок: генерали повинні обмінятися один з одним інформацією, виключивши помилкові дані. Точніше, вони повинні обмінятися інформацією про чисельність легіонів, вірних Візантії, і зробити висновки про чисельність легіону зрадників. Завдання передбачає, що при N кількості генералів зрадниками можуть виявитися N-1.

Принцип згоди полягає в тому, щоб усі вірні генерали в результаті обміну інформацією прийшли до однакового вирішення, проігнорувавши дані від генерала-зрадника.

Для того, щоб лояльні генерали досягли згоди, повинен існувати алгоритм який має гарантувати виконання двох умов[2]:

Умова А. Усі лояльні генерали вибирають однаковий план дій.

Лояльні полководці все зроблять так, як алгоритм каже, але зрадники можуть робити все, що завгодно. Алгоритм повинен гарантувати умову А незалежно від того, що роблять зрадники.

Лояльні полководці повинні не тільки досягти згоди, але й повинні погодитися на розумний план.

Умова Б. Невелика кількість зрадників не може змусити лояльних полководців прийняти поганий план.

Проблема Візантійських генералів. Генерал-командуючий повинен надіслати наказ своїм n - 1 генерал-лейтенантам таким чином:

Умова 1. Усі віддані лейтенанти підкоряються одному і тому ж наказу.

Умова 2. Якщо командуючий генерал є лояльним, то кожен вірний лейтенант виконує розпорядження, яке він надсилає.

Вирішення задачі візантійських генералів. В роботі [5] описано два алгоритму вирішення проблеми для усних та авторизованих повідомлень.

Усні повідомлення:

1. Кожне повідомлення, що надсилається, доставляється правильно.

2. Отримувач повідомлення знає, хто його відправив.

3. Відсутність повідомлення можна виявити.

Авторизовані повідомлення включають властивості усних повідомлень, але є також додаткові властивості такі, як:

4. а) Повідомлення, надіслані коректним процесом, не можуть бути підроблені і будь-які зміни в таких підписаних повідомленнях можна виявити.

б) Будь-хто може перевірити справжність підпису коректного процесу.

Алгоритм для усних повідомлень

Даний алгоритм буде працювати, якщо при m зрадників кількість лояльних генералів буде більше або дорівнювати $3m + 1$.

Даний алгоритм використовує функцію *Majority* (укр. Більшість), якщо більшість значень $v_i = v$, то $Majority(v_0 \ldots v_i) = v$, де v_i рішення i – ого генерала. Якщо значення *Majority*, не існує то рішення ВІДСТУПАТИ.

Алгоритм для усних повідомлень:

Крок 1: Генерал – командуючий відправляє своє значення v_i для усіх інших генералів.

Крок 2.1 Якщо $m = 0$, то кожен генерал використовує значення яке отримав від командира, або приймає рішення ВІДСТУПИТИ, якщо він не отримав значення від генерала – командуючого.

Крок 2.2 Якщо $m > 0$, то Якщо генерал отримав значення v_i від генерала – командуючого, то він виконує Крок 1 для n – 2 генералів.

Крок 3 Кожен генерал виконує функцію *Majority*($v_0 ... v_i$).

На рисунках 1.1 та 1.2 зображено роботу даного алгоритму коли зрадник генерал – командуючий та звичайний генерал відповідно, при $m = 1$ та $n = 4$.

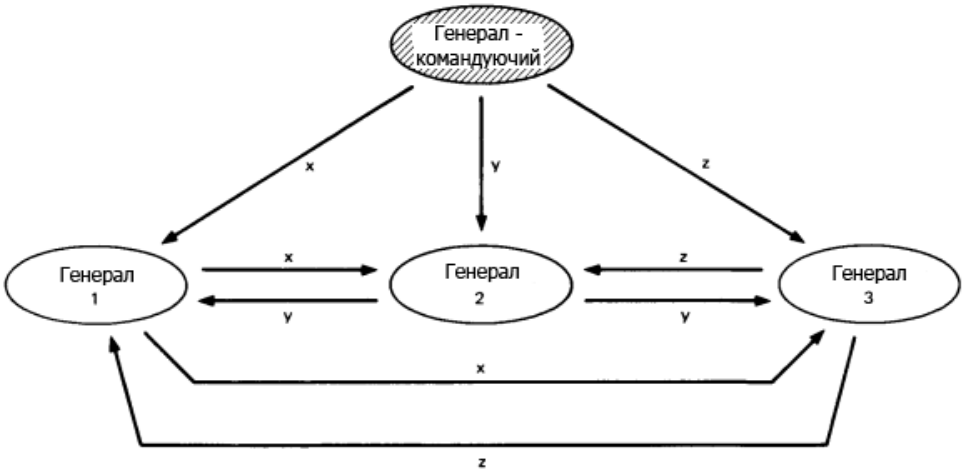

Рисунок 1.1 – Робота алгоритм для усних повідомлень коли зрадник генерал – командуючий

Рисунок 1.2 - Робота алгоритм для усних повідомлень коли зрадник генерал 3

1.4 Практичний протокол візантійської угоди (PBFT)

Даний алгоритм є формою скінченного автомату[6]. Сервіс моделюється як машина стану, який реплікується в різні вузли розподіленої системи.

Позначимо набір реплік як R, $R = \{0 \dots |R| - 1\}$. Для простоти ми припускаємо, що $R = 3f - 1$, де f - кількість “поганих” вузлів.

Репліки рухаються через послідовність конфігурацій, що називаються представленнями (англ. view). У представлені одна репліка є основною (англ. primary), а інша - резервною. Представлення нумеруються послідовно. Основною частиною представлення є репліка p така, що $p = v \bmod |R|$, де v – номер view. Функція *ChangeView* виконується коли основна репліка невдалася.

Алгоритм працює так:

1. Клієнт відправляє запит, на виклик сервісної операції, до основної репліки.
2. Основна реплікація відправляє запити до всіх резервних копій.
3. Репліки виконують запит і надсилають відповідь клієнту
4. Клієнт чекає відповідей f + 1 з різних реплік з однаковим результатом; це результат операції.

Як і для всіх методів реплікації скінченного автомату, ми пред'являємо дві вимоги до реплік: вони повинні бути детермінованими (тобто виконання операції в заданому стані і з заданим набором аргументів завжди повинно давати однаковий результат), і вони повинні стартувати з одного стану[7].

Враховуючи ці дві вимоги, алгоритм забезпечує властивість безпеки, гарантуючи, що всі несправні репліки узгоджують загальний порядок виконання запити, незважаючи на помилки.

Операції PBFT

Для операції фіксації блоку та досягнення прогресу вузли в мережі PBFT проходять три фази: попередня підготовкам (англ. pre-preparing), підготовка (англ. preparing), фіксація (англ. committing).

На рисунку 2.1 зображено дані три фази для 4 вузлів, де вузол під номером 3 є ненадійним.

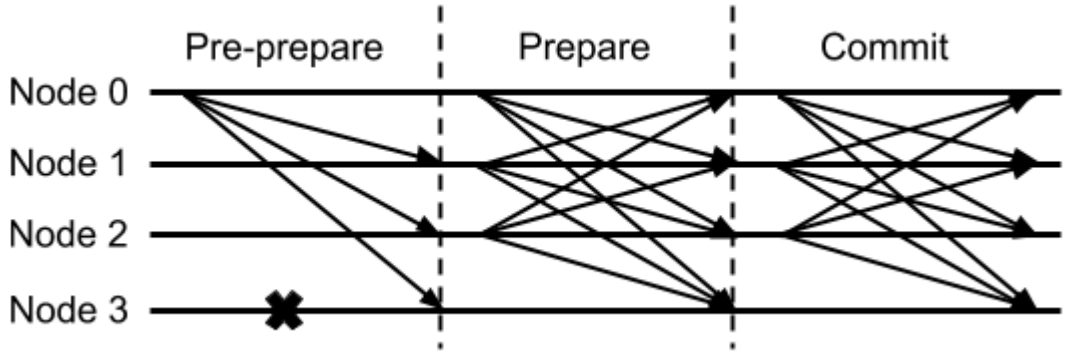

Рисунок 1.3 - Операції PBFT

Далі розглянемо кожну операцію окремо.

Попередня підготовка:

Для того, щоб почати роботу, основа для поточного стану створить блок та опублікує його в мереж[8]і; кожен із вузлів отримає цей блок та здійснить попередню перевірку, щоб переконатися, що блок дійсний.

Після того, як ключовий учасник опублікував блок в мережі, він передає попередньо підготовлене повідомлення всім вузлам. Попередньо підготовлені повідомлення містять чотири ключових поля: ідентифікатор блоку, номер блоку, номер стану основної реплікації і ідентифікатор основної реплікації. Коли вузол отримує заздалегідь підготовлене повідомлення від основного вузла, він перевірить його та додасть повідомлення до свого внутрішнього журналу. Перевірка повідомлень включає:

- перевірку цифрового підпису повідомлення,

- перевірку відповідності номера стану повідомлення поточному номеру.

Повідомлення заздалегідь готується як спосіб для основного вузла публічно схвалити даний блок і для мережі домовитись про те, який блок виконувати консенсус щодо цього порядкового номера. Щоб переконатися, що одночасно розглядається лише один блок, вузли не дозволяють більше ніж одне попередньо підготовлене повідомлення для заданого перегляду та порядкового номера.

Підготовка

Як тільки вузол отримав блок і попередньо підготовлене повідомлення для блоку, і блок, і повідомлення були додані до журналу вузла, вузол перейде до фази підготовки. На етапі підготовки вузол буде транслювати повідомлення про підготовку до решти мережі (включаючи себе). Підготовка повідомлень, як і попередньо підготовлені повідомлення, містить ідентифікатор та номер блоку, для якого вони є, а також номер і номер перегляду вузла.

Щоб перейти до наступної фази, вузол повинен почекати, поки він не отримає $2f + 1$ підготувати повідомлення, які мають однаковий ідентифікатор блоку, номер блоку та номер стану з різних вузлів. Дочекавшись $2f + 1$ відповідних підготовлених повідомлень, вузол може бути впевнений, що на цьому етапі всі належним чином функціонуючі вузли (ті, які не є несправними та не шкідливими) узгоджуються. Після того, як вузол прийняв потрібні $2f + 1$ відповідні підготовлені повідомлення та додав їх до свого журналу, він готовий перейти до фази фіксації.

Фіксація

Коли вузол переходить у фазу фіксації, він передає повідомлення про фіксацію всій мережі (включаючи себе). Як і інші типи повідомлень, повідомлення фіксації містять ідентифікатор та номер блоку, для якого вони створені, разом із номером перегляду вузла та ідентифікатором. Як і у фазі підготовки, вузол не може завершити фазу фіксації, поки не отримає $2f + 1$ відповідні повідомлення фіксації з різних вузлів. Знову ж таки, це гарантує, що всі несправні вузли в мережі погодилися зафіксувати цей блок, а це

означає, що вузол може безпечно зафіксувати блок, знаючи, що його не потрібно буде повертати[9]. Прийнявши необхідні повідомлення $2f + 1$ для фіксації у своєму журналі, вузол може спокійно фіксувати блок.

Після того, як первинний вузол закінчить фазу фіксації та здійснив блок, він запустить весь процес заново, створивши блок, опублікувавши його та транслюючи попередньо підготовлене повідомлення до нього.

Висновки до розділу 1

В першому розділі ми розібрали основні визначення, та терміни які нам будуть потрібні для подальшого розгляду проблеми масштабованості протоколів консенсусу в блокчейн, та для подальшої постановки задачі вирішення даної проблеми.

2 МЕТОДИ ВИРІШЕННЯ ПРОБЛЕМИ МАСШТАБОВАНОСТІ ПРОТОКОЛІВ КОНСЕНСУСУ

У даному розділі дипломної роботи буде проаналізовано можливі методи покращення масштабування протоколів консенсусу, та проаналізовано готові комплексні протоколи, методи і варіанти реалізацій даних методів для вирішення проблеми масштабованості протоколів консенсенсусу, а також проведений аналіз рішення масштабованості і аналіз безпеки варіантів реалізацій даних методів.

Для аналізу протоколів реалізованих з рішенням проблеми масштабованості взято протоколи Proteus та Checo.

2.1 Проблема масштабованості алгоритмів консенсусу що базуються на візантійській угоді

Для поглиблення у проблему масштабування протоколів BFT консенсенсусу. Розглянемо проблему з точки зору практичного протоколу BFT консенсенсусу.

Практичний протокол BFT консенсенсусу (англ. BFT Consensus Protocol, або скорочено PBFT) одна з найвідоміших реалізацій протоколів BFT консенсунсусу. Практичний протокол BFT консенсенсусу (PBFT) показав не спроміжність масштабування деяку, відносно невелику, кількість вузлів причиною цього є надлишковий обмін повідомленнями між між вузлами. Надлишковий обмін повідомленнями між вузлами[13] (англ. Communication overhead) спричиняє до нелінійного зменшення кількості транзакцій з часом.

Дана особливість протоколу накладає обмеження на його використання. Тобто при роботі по даному протоколу кожен комітет

(committee) у консорціумі (consortioum) може включати в себе (comprise) лише декілька десятків інститутів (institutions).

Більш того ймовірність зловмисного керування більш ніж третю частиною комітетів(Committee) досить висока при малих розмірах комітетів(Committee)[14]. Наша ціль покращити час на обмін даними між вузлами і збільшити толерантність до відмови(fault tolerance).

Шардинг блокчейна в значній мірі покладається на протоколи консенсусу основані на практичній візантійській помилці для досягнення консенсусу. Однак більшість протоколів даного виду обмежені в масштабованості або з точки зору розміру мережі (наприклад, кількості вузлів) або загальної пропускної здатності. Простір для їх вдосконалення є величезним. Ми будемо використовувати протоколи консенсусу основані на практичній візантійській помилці як приклад для пояснення масштабованості даних протоколів консенсусу. Оригінальний протокол PBFT вимагає $n = 3f + 1$ вузлів, щоб допустити до f візантійських помилок. Було в роботі [16] показано, що він не перевищує десяток вузлів через свою квадратичну складність зв'язку. Як правило, масштабування протоколів для BFT фокусується або на зменшенні кількості вузлів, необхідних для перенесення візантійських несправностей, або на зменшенні складності зв'язку протоколу, щоб дозволити більші розміри мережі.

Зменшення кількості вузлів. Щоб допустити f візантійські вузли, які можуть виділятися в системі кворуму, як PBFT, кворуми повинні перетинатися принаймні $f + 1$ вузлами [17]. Отже, якщо протокол BFT вимагає $n = 3f + 1$, його розмір кворуму становить щонайменше $2f + 1$. Менший n означає нижчі витрати на комунікацію, понесені при допущенні однакової кількості несправностей; це також означає, що для однакової кількості вузлів n мережа може переносити більш несправні вузли. Один із способів зменшити кількість вузлів - випадковим чином вибрати невеликий набір вузлів консенсусу як комітет, щоб запустити процес консенсусу. Менший комітет консенсусу може призвести до кращої пропускної

спроможності, оскільки менший комітет досягає більш високої пропускної здатності за рахунок менших витрат на комунікацію. Технологія шардингу зменшує процес консенсусу в межах одного шматка. Однак у цьому випадку безпека кожного фрагмента, наприклад, відношення кількості несправних вузлів до розміру осколка, буде головним питанням.

Інший спосіб зменшити кількість вузлів - це використання технік, щоб знизити n з $3f + 1$ до $2f + 1$. Ці методи в основному засновані на використанні зовнішніх компонентів (наприклад, надійного обладнання) або зменшенні моделей системи.

Зменшення складності зв'язку. Протокол PBFT сприймався як важкий протокол зв'язку. Існує давно міф про те, що BFT не піддається масштабуванню кількості учасників n, оскільки більшість існуючих рішень передбачає передачу повідомлення $O(n^2)$ навіть за сприятливих мережевих умов. Як результат, існуючі ланцюги BFT включають дуже мало вузлів. Навіть при зменшеному розмірі мережі PBFT все ще має складність зв'язку $O(n^2)$. Byzcoin [18] запропонував оптимізацію, в якій лідер використовує протокол колективного підписання (CoSi) [19] для об'єднання повідомлень інших вузлів в одне повідомлення, що підтверджується автентичністю. Роблячи це, кожному вузлу потрібно лише переслати свої повідомлення лідеру та перевірити сукупне повідомлення від останнього. Таким чином, уникаючи мовлення, складність зв'язку знижується до $O(n)$.

2.2 Метод вирішення проблеми масштабованості

Шардинг - це техніка досягнення горизонтальної масштабованості шляхом групування вузлів у кілька комітетів[10]. Вузли в межах одного групи запускають алгоритм візантійського консенсусу, щоб узгодити набір транзакцій, що належать цьому конкретному фрагменту. Міжрядковий протокол необхідний для транзакцій, що включають вузли з більш ніж

одного фрагмента. Кількість комітетів лінійно зростає відносно загальної обчислювальної потужності в мережі. Ця схема досягає горизонтальної масштабованості, оскільки якщо кожен комітет здійснює транзакції з однаковою пропускною здатністю, то додавання більшої кількості комітетів, природно, призведе до лінійного збільшення глобальної пропускної здатності. Прикладами блокчейн-систем, які використовують шардинг, є Elastico та OmniLedger.

Найбільшим обмеженням шардингу є те, що воно є оптимальним лише в тому випадку, якщо транзакції залишаються в одній гілці[11]. Насправді Elastico не може атомарно обробити міжрядкові транзакції. OmniLedger має протокол транзакцій між фрагментами, але вибрати хороший розмір комітету складно. Великий розмір комітету зробив би систему менш масштабованою, оскільки алгоритм візантійського консенсусу повинен управляти великою кількістю вузлів. Невеликий розмір комітету призведе до великої кількості транзакцій між шарами, що також перешкоджає масштабуванню. Крім того, використання дрібних осколків є ризиком для безпеки. Кожен осколок відрізняється відмовою, щонайменше, на третину розміру осколка[12]. Якщо загальна кількість шкідливих вузлів залишається зафіксованою, то, коли розмір осколка зменшується, швидше за все, осколок може вийти з ладу.

2.3 Збільшення рівня безпеки протоколів візантійського консенсенсу використовуючи довірене середовище виконання

Довірене середовище виконання (англ. – Trusted Execution Enviroment, або скорочено TEE).

Один з підходів удосконалення BFT протоколів полягає в припущені, що гібридна модель відмови (англ. hybrid failure model), в котрій деякі з компонентів являються довіреними і відмова компонентів відбувається лише при відмові компоненту через помилку виконання або збій у роботі самого

компонента, в той же час решта компонентів діють згідно Візантійської манери. Дана модель реалізована шляхом виконання довірених компонентів у довіреному середовищі виконання (TEE). Одна з ключових гарантій безпеки TEE виражається в забезпечені цілісності захищених компонентів, через що зловмисники не можуть реалізувати заміну виконання або якимось чином вплинути на виконання і спричинити зміну роботи захищених компонентів за заданим протоколом виконання. В даній роботі використовується “Intel Software Guard Extensions” (GSX) [23] для надання довіреного середовища виконання (TEE). Варто зазначити, що наша архітектура нашого спроектованого рішення дозволяє працювати також і з іншими постачальниками TEE, наприклад з “TrustZone”[24] або “Sanctum”[25].

Захист на основі закритої групи (англ. – Enclave protection)

“Intel SGX” надає підтримку захищеного середовища виконання у вигляді закритих груп. Закрита група в даному контексті визначається як адресний простір, захищений процесором, який доступний лише для процесів які належать закритій групі, тобто програми являються довіреними компонентами. Множині закриті групи можуть бути створені і запущені непривілейованими процесами запущеними від імені користувача. Закриті групи ізольовані один від одного, від операційної системи і від інших процесів. Ізольованість від ОС все ж, дозволяє використовувати деякі системні виклики такі як керування оперативною пам’ятю або запис/зчитування з жорсткого диску.

Протокол віддаленої атестації[26]

Користувач може завіряти чи конкретні довірені середовища виконання (TEE) коректно ініціалізовані і виконуються на віддаленому вузлі за допомогою протоколу віддаленої атестації. Після того як анклав який був сумнівний і потребував додаткового підтвердження користувача був ініціалізований, процесор обчислює геш початкового стану анклаву. Після

цього процесор підписує цей геш закритим ключем анклаву. На даному етапі користувач перевіряє підписаний геш і зрівнює його з відомим йому гешем.

Опечатування даних

Довірене середовище виконання(TEE) має можливість зберегти стан виконання на постійній пам'яті через механізм опечатування даних, що забезпечує можливість відновлення даних після відмови виконання процесу через збій. Для опечатування даних анклав спершу надсилає запит отримання унікального ключа побудованого закритим ключем анклаву і гешем початкового стану анклаву на процесор, потім шифрує дані і зберігає шифровані дані на постійну пам'ять. Даний механізм забезпечує, що шифровані дані можуть бути розшифровані лише закритою групою яка їх опечатала (зашифрував)[26]. Тим не менш, механізм відновлення опечатаних даних вразливий атакам у яких зловмисник надає закритій групі належним чином запаковані але попередні, не актуальні дані.

Криптографічні бібліотеки TEE

"Intel SGX"[27] також надає процесам закритої групи такі корисні функції як sgx_read_rand і sgx_get_trusted_time. Де перша функція надає доступ до генерації псевдовипадкових чисел з надійними криптографічними властивостями псевдовипадкових генераторів, а друга функція повертає різницю часу відносно деякої заданої опорної точки часу.

2.4 Аналіз протоколів з вирішеною проблемою масштабованості

Аналіз протоколу Proteus

Автори даного протоколу[22] стверджують, що продуктивність даного протоколу не обмежена порогом кількості відмов. Іншими словами на роботу протоколу не впливає кількість відмов, виявлених в мережі, і залишається

постійним (гарантує стабільну продуктивність) під час нормального виконання. Це дуже важлива і сильна характеристика, яка дозволяє протоколу забезпечувати стійкі гарантії продуктивності. Крім того даний протокол, за заявою авторів даного протоколу, забезпечує постійну затримку з точки зору критичної довжини шляху – це кількість повідомлень від пропозиції блоку до досягнення консенсусу.

Ці поліпшення досягаються випадковим вибором кількості реплік, які позначимо c з великого набору реплік із загальним розміром n, де n є звичайними репліками, де $c << n$. Даний алгоритм толерантний до кількості відмов не більших f, де $f < \frac{n}{3}$. Під час кожного виконання протоколу візантійський консенсус виконується спочатку серед реплік c, а не в загальному блокчейні. Блок, запропонований кореневим комітетом, буде підтверджений $n - f$ правильними репліками. Оскільки основний процес візантійського консенсусу буде виконуватися в кореневому комітеті розміром c, який набагато менший, ніж n, це дає нам кращі показники.

Складність досягання консенсусу в даному протоколі можна записати як $O(c^2 + cn)$, як для нормально режиму, так і для режиму при зміні скомпрометованого вузла. І для великих n, де $n >> c$ складність становить $O(cn)$.

Даний протокол може допустити до $\frac{2c}{3}$ візантійських збоїв у кореневому комітеті. Таким чином, кореневий комітет є більш стійким, ніж типовий випадок, який допускає менше $\frac{c}{3}$ відмов вузлів. У разі відмови $\frac{2c}{3}$ вузлів виконується зміна кореневого комітету. Це ще один унікальний аспект даного протоколу, оскільки зміна представлення викликає заміну кореневого комітету, тоді як інші протоколи на основі BFT замінюють лише первинний вузол.

Системна модель протоколу Proteus

В протоколі Proteus використовується візантійська модель помилок. Відповідно до цієї моделі, сервери та клієнти можуть відхилятися від своєї

нормальної поведінки довільними способами, що включає збої обладнання, помилки програмного забезпечення та іншу шкідливу поведінку. Даний протокол може терпіти до f візантійських реплік, де загальна кількість реплік в мережі n така, що $n = 3f + 1$. Розмір кореневого комітету c може бути обраний на основі вимоги гарантії безпеки (P_f) для комітету що виконують візантійський консенсус. Ця модель також передбачає, що репліки не зможуть зламати геши, шифрування та підписи, стійкі до колізій. Також припускається, що всі повідомлення, надіслані репліками, підписані. Для забезпечення життєдіяльності в асинхронній мережі використовується тайм-аут, щоб розмістити верхню межу часу генерації блоку.

Принцип роботи протоколу Proteus

У Proteus автори перенесли задачу досягання консенсусу до кореневого комітету розміром c. Кореневий комітет працює з налаштованим алгоритмом на основі візантійського консенсусу. Регулярні репліки, які не є членами кореневого комітету, просто перевіряють результат консенсусу, тобто запропонований блок. Якщо репліки кореневого комітету $\frac{2c}{3} + 1$ узгоджують блок, запропонований кореневим комітетом, і загалом $2f + 1$ (кореневий комітет плюс звичайні репліки) погоджуються на запропонований блок, тоді блок здійснюється і буде доданий до блокчейну. Після відмови протоколу консенсус додати новий блок, увесь кореневий комітет змінюється. Звичайне виконання протоколу можна розділити на такі етапи:

1) Протокол консенсусу в кореневому комітеті успішно генерує блок і пропонує йому регулярні репліки через трансляцію.

2) Після отримання блоку регулярні репліки перевіряють чи підписаний блок $2c / 3 + 1$ репліками від кореневого комітету.

3) Якщо блок дійсний, кожна регулярна репліка підписує блок і надсилає назад підпис кореневому комітету.

4) Після отримання $2f + 1$ підписів від звичайних реплік, а також членів кореневого комітету кожен член кореневого комітету здійснює фіксацію

блоку і передає доказ про прийняття ($2f + 1$ підписів) блоку до регулярних реплік.

5) Після отримання доказів кожна звичайна репліка здійснює фіксацію блоку і додає його до блокчейну.

А. Вибір членів кореневого комітету

Члени кореневого комітету вибираються випадковим чином із загальної кількості реплік n. Припустимо, що з n вузлів f скомпрометовано, де $f < n / 3$. Розмір c - це заздалегідь визначене число, яке вказує середній розмір кореневого комітету. Нехай V – сукупність вузлів у мережі, $|V| = n$. Ми можемо записати $V = A \cup B$, де A – сукупність коректних вузлів, а B – скомпрометовані вузли, таких, що $|A| = n - f$ і $|B| = f$. Нехай C позначає кореневий комітет такий, що $|C| = c$. Ми припускаємо, що C утворюється випадковим і рівномірним підбором набору c вузлів з n. Тому кількість можливих способів вибору будь-якого конкретного набору вузлів можна порахувати, як: $\binom{n}{c}$. Ймовірність вибрати саме a чесних вузлів та b не чесних вузлів для C, де $a+b=c$, рахується як:

$$\frac{\binom{n-f}{a}\binom{f}{b}}{\binom{n}{c}} \quad (2.1)$$

Таким чином, ймовірність у кореневому комітеті більше $2c / 3$ не чесних вузлів b буде сумою всіх ймовірностей від $2c / 3 + 1$ до c:

$$P_f = \begin{cases} \sum_{b=2c/3+1}^{c} \frac{\binom{n-f}{a}\binom{f}{b}}{\binom{n}{c}}, & \text{якщо } f > 2c/3 \\ 0, & \text{в інших випадках} \end{cases} \quad (2.2)$$

Безпека даного алгоритму не залежить від кількості шкідливих / несправних реплік у кореневому комітеті. Але для життєдіяльності нам потрібна хоча б одна чесна / правильна репліка, щоб бути членом комітету $2c / 3 + 1$, який генерує блок.

Для роботи кореневого комітету він випадково вибирається з набору n реплік, а розмір c такий, що ймовірність відмови, що має менше $c / 3$ чесних реплік, є незначною. Якщо кореневий комітет не може генерувати дійсний

блок, він замінюється іншим випадковим чином обраним комітетом. Заміна кореневого комітету новообраним комітетом називається зміною кореневого комітету. У даному протоколі зміна подання відрізняється від типової зміни перегляду BFT-протоколу, де замінюється лише первинна репліка, тоді як в нашому протоколі весь кореневий комітет замінюється. За звичайних обставин це рідкісна подія, оскільки ймовірність наявності більше ніж ***c / 3 - 1*** візантійських реплік у кореневому комітеті є низькою, і коли більшість членів кореневого комітету чесні, протокол працює протягом достатнього періоду без зміни кореневого комітету.

Б. Випадки відмови

Виходячи з нашого припущення, в кореневому комітеті є щонайменше c / 3 чесних репліки. Це призводить до можливості існування таких випадків відмови:

1) Кількість шкідливих реплік у кореневому комітеті перевищує ***c / 3***: Якщо кількість шкідливих реплік становить щонайменше ***c / 3 + 1***, і вони вирішили вести себе зловмисно, блок не буде генерований, так як не буде можливо зібрати ***2c / 3 + 1*** підписи для блоку. Таким чином, розпочнеться процес зміни кореневого комітету.

2) Первинна репліка в кореневому комітеті є шкідливою: Якщо первинна репліка є шкідливою, кореневому комітету може не вдатися генерувати блок успішно, оскільки первинна репліка може просто не ініціювати пропозицію блоку. У такому випадку час очікування призведе до зміни кореневого комітету.

3) Первинна репліка в кореневому комітеті, а також більше чим ***(2c / 3) - 1*** член кореневого комітету є зловмисним: У цьому випадку зловмисні репліки в кореневому комітеті можуть вступати в змову з шкідливою первинною реплікою, щоб змусити чесні репліки в кореневий комітет приймає різні блок-пропозиції. Але оскільки чесні репліки в кореневому комітеті транслюють пропозиції блоку (поряд із повідомленнями

членів кореневих членів для блоку), інша репліка виявить цю невідповідність, і це призведе до зміни кореневого комітету.

Аналіз протоколу CHECO

Архітектура системи CHECO

CHECO побудований на блокчейні TrustChain, а точніше, автори, даного протоколу, зробили його модифікацію, та назвали її Розширений TrustChain (Extended TrustChain), а також використовує три протоколи, які можна побачити на рисунку 2.2.

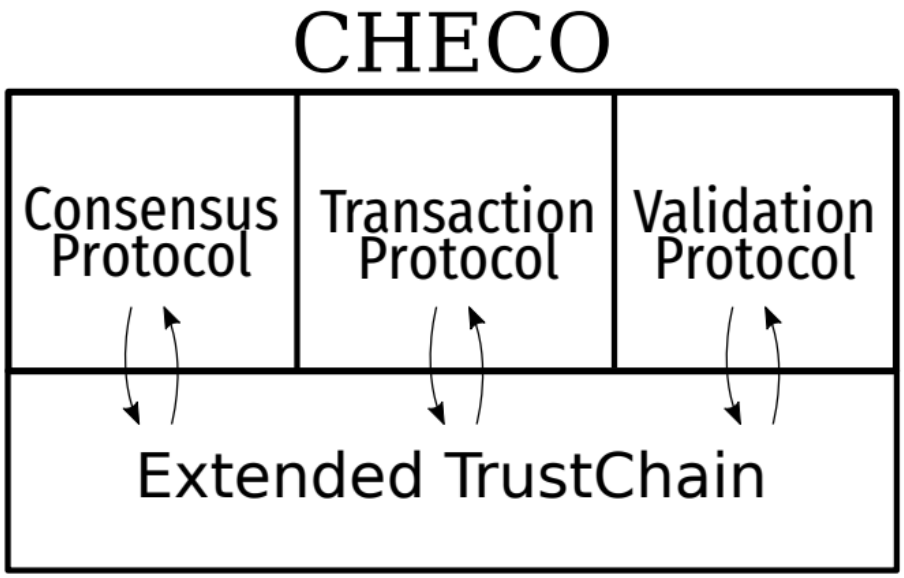

Рисунок 2.2 - Системна архітектура CHECO [20]

Протоколи мають чітку роль, що видно з їх назв. Вони працюють одночасно і не взаємодіють між собою. Єдина синхронізація відбувається на шарі Extended TrustChain, де всі протоколи читають і записують з нього.

TrustChain та Розширений TrusChain

Докладний опис TrustChain можна отримати в роботі [21]. Тут буде написано лише основні моменти, та різницю між TrustChain та розширеним TrustChain

У TrustChain кожен вузол має персональний геш-ланцюг. Спочатку ланцюг містить лише твірний блок, генерований самими вузлами. Коли вузол A хоче зробити транзакцію (TX) з B, новий блок TX генерується як для A, так і для B і додається до їх відповідних ланцюгів. Блок TX повинен мати дійсний геш-покажчик, що вказує на попередній блок, і посилання на його пару в ланцюжку B. Приклад наведено на рисунку 2.3.

Якщо кожен вузол слідує правилам TrustChain і ми розглядаємо лише геш-покажчики, а не посилання, то кожен ланцюжок ефективно формує окремо звязний список. Однак, якщо вузол порушує правила, випадково чи зловмисно, це призводить до утворення розгалуження в ланцюгу блоків. Тобто може бути більше одного блоку TX з геш-покажчиком, що спрямований на той самий блок. На малюнку 2.3, вузол b (в середньому ланцюзі) створив два блоки TX, які обидва вказують на $t_{b,5}$. Дану ситуацію можна розглядати як подвійне витрачання, де валюта, накопичена до $t_{b,5}$, витрачається двічі.

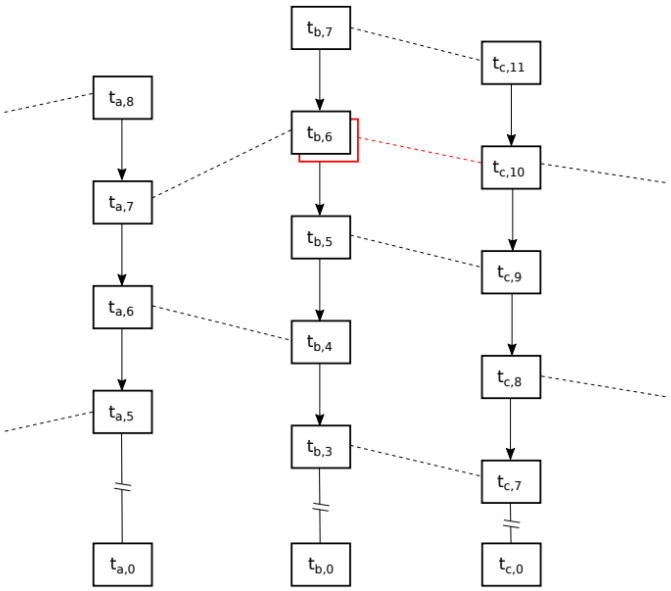

Рисунок 2.3 - Структура TrusChain

Розширений TrustChain представляє новий тип блоку - блок контрольної точки (CP). Основна функція блоку CP полягає в поданні стану вузла у вигляді геш-покажчика. Колекція блоків CP з усіх вузлів представляє стан всієї системи. Тому вузли повинні досягти консенсусу щодо блоків CP, використовуючи якийсь протокол консенсусу, щоб мати інформацію про стан машини. Дані блоки можна побачити на рискунку 2.4.

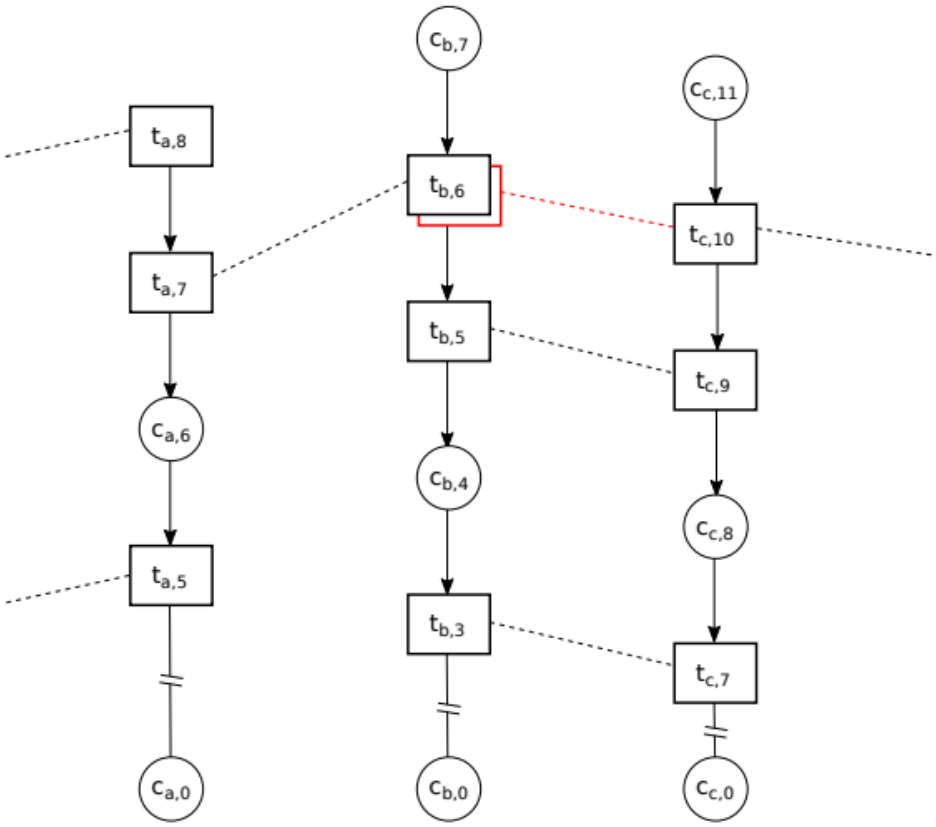

Рисунок 2.4 - Структура розширеного TrusChain

Протокол консенсусу

Протокол консенсусу можна розглядати як техніку безкінечного виконання візантійського алгоритму консенсусу, починаючи нове виконання відразу після завершення попереднього. Вузли створюють нові блоки CP в кінці кожного виконання. Такий підхід необхідний, оскільки блокчейн-системам завжди потрібно досягти консенсусу щодо нових значень, запропонованих вузлами в системі, або CP-блоками в нашому випадку.

Складність зв'язку візантійських алгоритмів консенсусу, як правило, зростає поліноміально з кількістю вузлів, що забороняє нам запускати його у

великій мережі. Наприклад, на одну з останніх робіт, протестовану до 104 вузлів, і алгоритм на виконання знадобився майже 6 хвилин [29]. Таким чином, на початку виконання кожного алгоритму візантійського консенсусу ми довільно обираємо набір вузлів, які називаються фасилітаторами, - щоб збирати блоки CP з кожного іншого вузла і використовувати ці блоки як вхід до алгоритму візантійського консенсусу, а потім запускати його. Після завершення алгоритму фасилітатори виводять набір блоків CP, які ми називаємо результатом консенсусу, який потім поширюється в мережу. Використовуючи результат, вузлам дозволяється створювати нові блоки CP, і тоді починається наступне виконання алгоритму.

Протоколи транзакції та перевірки

Протокол транзакцій - це простий протокол запиту та відповіді. Вузли обмінюються одним раундом повідомлень і створюють нові блоки TX у відповідних ланцюгах.

Сам по собі протокол консенсусу та транзакцій не забезпечує механізм виявлення зловмисної поведінки, наприклад, підробки TrustChain. Таким чином, нам потрібен протокол перевірки для протидії такій поведінці. Коли вузол хоче підтвердити одну зі своїх транзакцій, він просить контрагента про узгоджений фрагмент транзакції. Який є фрагментом ланцюга контрагента, що починається і закінчується блоками CP, але містить блок TX, що належить до цієї транзакції, і блоки CP повинні бути консенсусними. Після відповіді контрагента, вузол перевіряє, чи відповідають блоки блоків CP в консенсусі та в деяких інших умовах. Угода діє, якщо ці умови виконані. Інтуїтивно це працює, тому що важко (оскільки криптографічно захищена геш-функція стійка до колізій) створити інший ланцюг, який починається і закінчується тими ж двома блоками CP.

Висновки до розділу 2

У другому розділі дипломної роботи було проаналізовано готові комплексні протоколи Proteus та Checo які були створені з акцентом на покращення масштабованості Візантійського консенсусу.

Були проаналізовані наступні існуючі методи покрашення масштабованості у протоколах BFT консенсенсу для блокчейн мереж: метод зменшення вузлів, метод оптимізації повідомлень обміну між вузлами і повідомленнями консенсенсу і метод у якому рішення проблеми масштабування полягала в модифікації алгоритму шардингу.

3 МЕТОДИ ВИРІШЕННЯ ПРОБЛЕМИ МАСШТАБОВАНОСТІ

3.1 Вирішення проблеми за допомогою шардінгу

Формування механізму шардингу

Задача формування шардингу у блокчейн системі більш комплексна ніж у розподіленій базі даних[33]. По перше, вузли потрібно присвоювати комітетам без упереджень і у випадковій манері. По-друге, розмір кожного комітету повинен бути ретельно підібраний для того, щоб досягти компромісу між ефективністю та безпекою. І, нарешті, присвоювання комітетів повинно виконуватися з деяким періодом для запобіганню адаптивних атак які сфокусовані на компрометацію більшості вузлів у комітеті. У цьому розділі представлений один із можливих підходів у вигляді експлуатації додаткових функцій TEE для вирішення даної задачі.

Генерація випадкового розподілу

Для безпечного формування шарду потрібне неупереджене випадкове число rnd для призначення вузла до комітету. За допомогою rnd, вузли отримують своє назначення до комітету, вичислюючи випадкову перестановку π з усіх можливих перестановок $[1:N]$ згенерованих за допомогою rnd. Потім π ділиться на приблизно однакового розміру чанки, кожен з яких представляє усі вузли в одному комітеті.

В даному випадку, буде використана TEE для ефективного отримання rnd у розподіленому Візантійському просторі шляхом додавання до кожного вузла анклав, який повертає випадкове значення – $RandomnesBeacon$. Відповідно до попередніх робіт, робиться припущення, що мережа синхронізації має визначену затримку Δ під час процедури генерації випадкового значення.

В даній роботі приклад реалізації шардінгової блокчейн системи працює в епохах. Кожна нова епоха відповідає призначенню нового вузла до комітету. На початку кожної епохи, кожен вузол викликає значення анклаву $RandomnesBeacon$ з номером епохи e. Анклав генерує два випадкових значення q і rnd, використовуючи два незалежні виклики функції $sgx_{read_{rand}}$. Потім анклав повертає підписаний сертифікат який містить у собі (e, rnd) тоді і тільки тоді, коли $q = 0$. Сертифікат розсилається усім вузлам у мережі. Після часу Δ, вузли фіксуються у найнижчому rnd який був отриманий для епохи e, і використовує це значення для обчислення комітету до якого призначати вузли.

Анклав налаштований таким чином, що його можна викликати лише один раз за епоху, що не дозволяє зловмиснику вибірково відкидати вихідні значення анклаву з метою зміни кінцевого випадкового значення rnd. Якщо вузол не зміг отримати жодного повідомлення після часу затримки Δ, що відбувається у разі коли вузол приймає (e, rnd) з свого анклаву, то e інкрементиця і процес повторюється знову.

Ймовірність повторення процесу: $P_{repeat} = (1 - 2^{-l})^N$, де l кількість бітів q. Це значення може бути змінено для досягнення бажаного компромісу між P_{repeat} і надлишкового обміну повідомленнями між вузлами представлену у вигляді: $O(2^{-l}N^2)$. Наприклад, коли $l = \log(z)$ для деякої константи z, $P_{repeat} \approx e^{-1}$ і кількість обмінів повідомленнями $O(N)$.

Розмір комітетів

Так як призначення до комітету визначається випадковою перестановкою з усіх можливих перестановок $\pi \in [1 : N]$ заданою значенням rnd, призначення до комітету може розглядатися як випадкова вибірка без заміни. Звідси ми можемо обчислити ймовірність відмови комітету (коли комітет містить у собі більше ніж f візантійський вузлів) використовуючи гіпергеометричний розподіл. Нехай X є випадковою змінною, яка представляє кількість візантійських вузлів, присвоєних комітету

розміром n, враховуючи загальний розмір мережі N вузлів, серед яких не більш ніж $F = sN$ вузлів є візантійськими. Тоді ймовірність відмови комітету (ймовірність, що безпека скомпрометована) така:

$$Pr[X \geq f] = \sum_{x=f}^{n} \frac{\binom{F}{x}\binom{N-F}{n-x}}{\binom{N}{n}} \quad (2.3)$$

Ймовірність відмови комітету можливо зменшити до незначної, шляхом налаштування розміру комітету згідно формули (1). Якщо $f \leq \frac{n-1}{3}$ (у випадку PBFT), за наявності 25% протидіючої потужності, кожен комітет повинен містити 600+ вузлів, щоб зменшити ймовірність відмови комітету до мінімальної. При використані протоколу AHL+ кожен комітет може витримати до $f = \frac{n-1}{2}$ відмов, звідси групи можуть бути значно меншими: $n = 80$, для $Pr\left[X \geq \frac{n-1}{2}\right] \leq 2^{-20}$.

Не великі розміри комітетів призводять то покращення швидкості виконання через дві причини. Перша – групи можуть досягнути більшої пропускної здатності через зменшення надлишкового обміну повідомленнями між вузлами. Друга – решта комітетів в мережі можуть збільшити пропускну здатність по причині меншого робочого навантаження.

Переналаштування груп

Атака адаптивного типу може скомпрометувати не відмовні групи або чесні вузли, хоча така компрометація вузла досить складна по часу[34]. Як результат має місце наступне твердження: періодичне перепризначення або переналаштування груп, які переставляють вузли серед комітетів є достатньою умовою для захисту від даного типу атак. Реконфігурація груп виникає за кожної епохи e. Після епохи $e - 1$, вузли отримують випадково згенероване значення rnd. Вузли обраховуюь нове призначення до комітету для епохи e залежно від значення rnd. Вузли які були перепризначені до комітету називатимемо *перехідними вузлами*. Період під час якого перехідний вузол направляється до нового комітету називатимемо періодом *зміни епохи*.

Під час зміни епохи, перехідні вузли спочатку перестають обробляти запити їх старих комітетів, потім починають отримувати стани їх нових комітетів від нинішніх членів вузлів відповідних комітетів. Тільки після завершення отримання стану, вузли офіційно приєднуються до нового комітету і починають обробку транзакцій. У період зміни епохи перехідні вузли не беруть участі у протоколі консенсенсусу ні у старих, ні у нових комітетів. Звідси слідує, що реконфігурація, при якій усі вузли відразу являються перехідними не бажана, бо це призводить до непрацездатності системи на період зміни епохи усіх вузлів.

Оптимальний підхід тоді, коли усі вузли переходили до інших комітетів партіями. Зокрема, для кожного комітету, не більш ніж B вузлів переходять до нового комітету за епоху e. Порядок переходу вузлів визначається випадковим значенням rnd. Далі буде розглянуто вплив B на безпеку та життєвий цикл шардінгового блокчейну.

3.2 Вирішення зменшенням кількості вузлів.

Якщо протокол консенсенсусу може перешкоджати двозначності Візантійських вузлів (тобто видавати конфліктні твердження різним вузлам), то протокол може бути толерантним до $f = \frac{N-1}{2}$ не двозначних Візантійських відмов (Byzantine failures) з усіх N вузлів[28]. Двозначність може бути усунена шляхом виконання усього протоколу консенсенсусу всередині довіреного середовища виконання(TEE), таким чином зменшити модель відмови від Візантійської (BFT) до відмови збою виконання.[35].

Одна з причин чому, на мою думку, даний підхід не буд реалізований це те що використання даного підходу спричиняє до значного збільшення довіреної обчислювальної бази (англ. trusted computing base, скорочено TCB). Масштабні довірені обчислювальні бази не бажані для супроводу належного

рівня безпеки через її комплексність, або навіть унеможливлює її через неможливість проведення аналізу безпеки коду, що несе у собі збільшення потенційних вразливостей і загроз[36].

Замість даного очевидного способу був використаний протокол автором якого є Бюн-Гон Чун [28], даний протокол для виключення двозначності використовує абстракцію представлену у вигляді журналу довіреного коду, яка називається «атестована, тільки для додавання пам'ять» (англ.. – attested append-only memory). Журнал підтримується і оперує зсередини довіреного середовища виконання (TEE), чим забезпечується захист від підміни запису операцій до журналу зловмисником. Цей протокол був застосований поверх Hyperledger Fabric v0.6 з використанням довіреного середовища виконання Intel SGX далі, зв'язка цих протоколів називатиметься Attested HyperLedger (AHL).

Attested HyperLedger підтримує різні журнали для різних типів повідомлень консенсусу (наприклад, pre-prepare, prepare, commit). Перед тим як відправляти нове повідомлення, кожен вузол повинен додавати дайджест повідомлення у відповідний журнал. Доказ виконання цієї операції містить підпис-сигнатуру, створену довіреним середовищем виконання (TEE) і цей підпис включається в саме повідомлення. Attested HyperLedger вимагає, щоб усі валідні повідомлення супроводжувались таким доказом.

Кожен вузол отримує і завіряє $f + 1$ повідомлення підготовки – prepare, перед тим як перейти до фази доручення повідомлення – commit, далі кожен вузол відправляє $f + 1$ commit повідомлень і вже після цього виконується сам запит. Attested HyperLedger періодично опечатує журнали повідомлень (запаковує за допомогою секретного ключа і гешу стану довіреного середовища виконання (TEE)) і записує опечатані журнали повідомлень на носій постійної пам'яті.

Але як було вказано раніше, даний механізм резервування не забезпечує захист від атак типу відкату запакованого журналу повідомлень

[37]. Автор даної роботи все ж описує один із варіантів реалізації механізму захисту віл даної загрози, але для огляду в дипломній роботі це не важливо.

3.3 Вирішення з допомогою оптимізації обміну повідомлень між вузлами та повідомлень консенсусу

Тестування роботи протоколу AHL[28] показало, що авторам даного протоколу все ж не вдалось досягнути достатнього рівня масштабованості. З результатів роботи цих авторів видно, що в їхньому протоколі велика кількість консенсусних повідомлень падає, як наслідок це призводить до низької пропускної здатність, коли розмір блокчейн мережі збільшується.

Пізніше у роботі[28] було реалізовано дві оптимізації для покращення обміну повідомлень між вузлами системи і протокол з цми доданими оптимізаціями в даній роботі називається Attested HyperLedger +- (AHL+).

За допомогою першої оптимізації була змінена і поліпшена робота протоколу HyperLedger, шляхом використання однієї мережевої черги для передачі консенсусу і запитів повідомлень. Зокрема було розділено попередню чергу повідомлення на 2 відокремлених канали, кожен з яких використовується для різного типу повідомлень. Повідомлення які отримані з мережі містять метадані які визначають тип цих даних і перенаправляють повідомлення у відповідні канали.

Розбиття в даній роботі черги повідомлення на 2 канали, простий і ефективний метод який запобігає переповнення черги цими запитами повідомлень, переповнення черги спричиняє відкидання пакетів які містять консенсуси.

Також у протоколі консенсусу AHL виникає не бажана поведінка виконання, коли репліка отримує запит користувача, по специфікації реалізації PBFT протоколу, цей запит надсилається усім вузлам відразу[25]. Така поведінка не обов’язкова і надмірна бо при отримані запиту ведучим

вузлом, цей вузол знову надсилає запит усім вузлам під час фази попередньої підготовки. Пізніше дана проблема була вирішена.

Варто зазначити і те що, автори протоколу AHL+ додали ще одну оптимізацію запозичену у Bizcoin[37], де ведучий вузол збирає та агрегує повідомлення інших вузлів в одне автентифіковане повідомлення. Далі кожен вузол пересилає свої повідомлення ведучому вузлу і завіряє агреговане повідомлення від попереднього вузла. В результаті даної модифікації, надлишковий обмін повідомленнями знижений до прийнятного рівня часової складності O(N).

Протокол AHL+ з цими модифікаціями автори назвали Attested HyperLedger Relay (AHLR). Підсумовуючи кроки які були зроблені та описані вище, протокол AHLR реалізований поверх протоколу AHL який за допомогою журналу повідомлень в довіреному середовищі виконання завіряє і агрегує повідомлення. Якщо при виконані протоколу AHLR було отримано $f + 1$ валідних, підписаних повідомлень для запиту req, під час фази виконання p на ітерації косенсенсусу o, журнал у TEE видає підтвердження, що відбувся кворум(quorum) для трійки (req, p, o)

3.4 Порівняння результатів і аналіз безпеки методів вирішення проблеми масштабованості

Результати варіанту реалізації методів зменшення кількості вузлів і оптимізації повідомлень

Результати методів зменшення кількості вузлів і оптимізації повідомлень базуються на результатах протоколи AHL, AHL+ і AHLR, які являються модифікацією протоколу PBFT – Hyperledger v0.6. AHL має лише одну відмінність від протоколу Hyperledger, яка полягає в основному поліпшені безпеки – журнал операцій протоколу всередині TEE. AHL+ і AHLR використовують метод оптимізації шляхом зменшенням кількості

вузлів. AHL+ до того ж використовує оптимізацію обміну повідомлень між вузлами, а AHLR в свою чергу використовує і оптимізацію обміну повідомленнями між вузлами, і оптимізацію повідомлень консенсенсу.

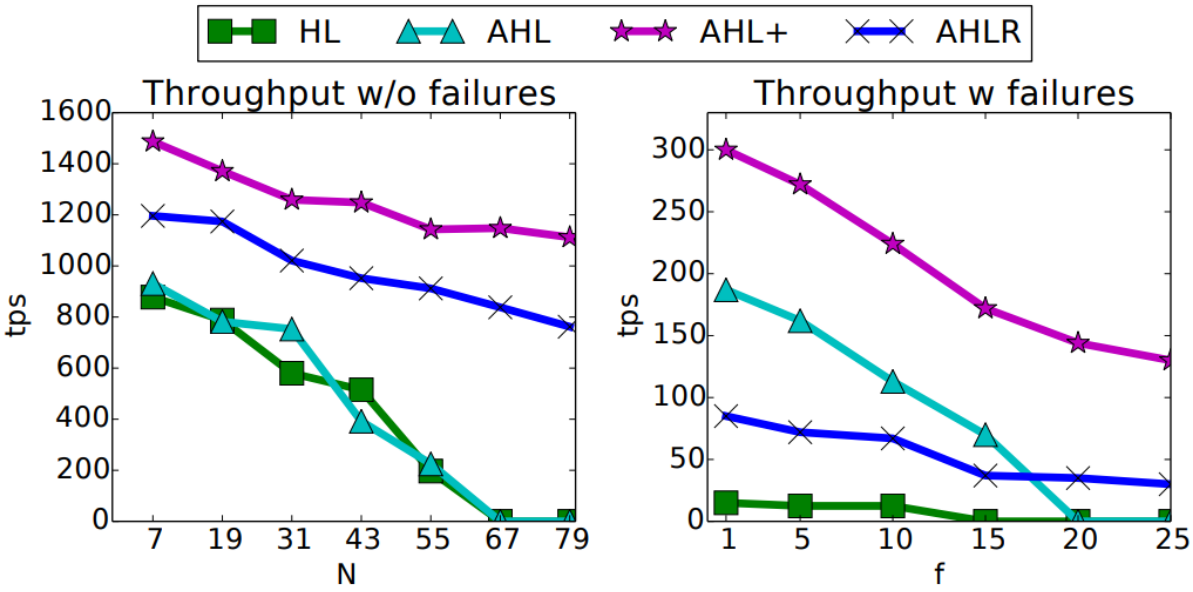

Рисунок 2.5 – Графіки продуктивності роботи протоколів з використанням даних методів на локальному кластері

На рисунках 2.5 представлено порівняння продуктивності протоколу AHL+, AHLR з PBFT протоколом у реалізації Hyperledger. Графіки відображають пропускну здатність транзакцій з збільшенням кількості вузлів N на локальному кластері і на платформі хмарних обчислень, де має місце затримка передачі даних.

Пропускна здатність протоколу AHL майже така ж як і у HL, але при однаковій кількості вузлів N, AHL протокол здатність витримувати більше відмов. Обидва протоколи показують нульову пропускну здатність при кількості вузлів $N > 67$ як на локальному кластері так і на хмарному.

Ми можемо бачити, що система легко може завмирає при великих значеннях N на фазі зміни представлення вузлів. І кількість змін вузлів веде до збільшення Візантійських відмов. Також з графіка видно, що і AHL+, і AHLR можуть оброблювати до 700 транзакцій в секунду на локальному

кластері і більш ніж 200 у хмарі. AHL+ показує стабільно кращий результат чим AHLR, хоча AHL+ має складність обміну повідомленнями $O(N^2)$ в порівняні з AHLR $O(N)$. Звідси можна зробити висновок, що дана проблема зв'язана з обміном повідомленнями, коли відбувається відмова вузла і наступний вузол не може агрегувати і розіслати повідомлення вузлам за заданий час і система намагається змінити представлення вузлів, що займає досить багато часу.

Результати варіанту рішення проблему у блокчейн шардінгу

За варіант рішення і виміри протоколу з спробою покращити проблему масштабування шляхом блокчейн шардінгу взято протокол блокчейн шардінгу з статті яка згадувалася в вступі цього розділу.

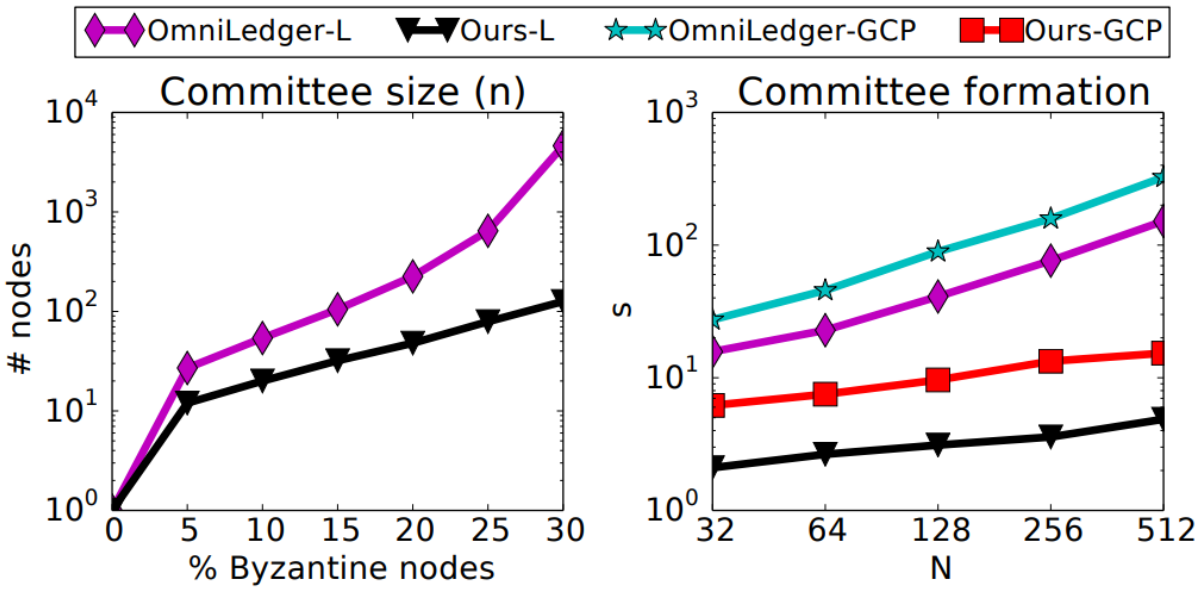

Рисунок 2.7 – Оцінка продуктивності формування груп даного протоколу

Графік на рисунку 2.7 порівнює продуктивність протоколу формування груп з іншими протоколами по таким параметрам як розмір комітетів і час на формування шарду. З збільшенням Візантійських відмов, OmniLedger потребує експоненціального збільшення комітів. Так як даний протокол формування груп побудований на AHL+, розмір комітетів залишається на 2 порядка менший відносно його аналогів.

Аналіз безпеки рішення масштабованості шляхом зменшення кількості вузлів і оптимізації повідомлень

Проаналізуємо рівень безпеки використання даних методів у алгоритмах AHL+, AHLR. AHL+ і AHLR обидва використовують метод оптимізації шляхом зменшенням кількості вузлів. AHL+ використовує оптимізацію обміну повідомлень між вузлами, а AHLR в свою чергу використовує і оптимізацію повідомлень консенсенсусу.

Журнал довірених операцій який використовується у Attested HyperLedger, доволі надійно захищений, бо підписаний за допомогою приватного ключа генерованого в середині анклаву довіреного середовища виконання Intel SGX.

Оскільки зловмисник не має змоги підробити сигнатури журналу викликів операцій, звідси неможливо спричинити двозначність (англ. equivocation) повідомлень Візантійських вузлів.

Якщо в роботі протоколу було отримано не більш ніж $f = \frac{N-1}{2}$ не двозначних Візантійських відмов, протокол AHL гарантує безпеку незалежно від умов стану мережі і незалежно від життєдіяльності при виконанні у частково синхронній мережі[38].

Так як у протокола AHL+ було додана тільки оптимізація обміну повідомлень між вузлами і не були внесені зміни які стосуються повідомлень консенсенсу, звідси робимо висновок, що рівень безпеки AHL+ еквівалентний рівню безпеки AHL.

AHLR оптимізує обмін повідомленнями між вузлами при умові, що не було змінено представлення вузлів, AHLR використовує такий ж самий протокол зміни представлення як і AHL. Оскільки агрегація повідомлень виконується у анклаві довіреного середовища виконання, AHLR має такі ж самі гарантії безепеки як і AHL.

Аналіз безпеки шардінгового блокчейну

Нехай k кількість груп, де кожна група представляє частку глобального стану блокчейна. Реконфігурація групи суттєво змінює набір вузлів які обробляють запити для кожного з k груп. Розглянемо групу sh і позначимо комітет, обробляючий шард sh в епоху $e-1$ часом C_{e-1} і епоху e часом C_e. Так як B вузлів змінюють комітет протягом часу C_{e-1}, і існує $\frac{n}{k}$ вузлів які протягом часу C_{e-1} очікують залишитися з часом C_e, то $\frac{n(k-1)}{k*B}$ проміжних комітетів обробляє щард sh під час періоду зміни епохи.

Перехід B вузлів не порушує безпеку група sh, тому що кількість Візантійських вузлів не збільшується. З іншої сторони, коли нові вузли B переходять до нового комітету, кількість візантійських вузлів в проміжних комітетів може збільшити розмір комітету до рівня при якому ймовірність відмови комітету буде висока, а ймовірність відмови основний фактор рівня захищеності комітетів. Так як перехідні вузли обираються випадково в залежності від значення rnd, то ймовірність відмови проміжного комітету відповідає формулі (1). Очікується, що існує $\frac{n(k-1)}{k*B}$ таких проміжнх комітетів у часовому проміжку з C_{e-1} до C_e. Використаємо булеву нерівність щоб оцінити ймовірність того, що безпека шарду sh на належному рівні під час епохи переходу: $Pr_{faulty} \leq \sum_{i=1}^{\frac{n(k-1)}{k*B}} \sum_{x=f}^{n} \frac{\binom{F}{x}\binom{N-F}{n-x}}{\binom{N}{n}}$ (2)

Наприклад, при $n = 80, f = \frac{n-1}{2}, k = 10$ груп і вузлів $B = \log(n) = 6$, $Pr_{faulty} \approx 10^{-5}$. Виходячи з формули (2), можна мати B перехідних вузлів для компромісу між життєздатністю і безпекою системи під час переходу епохи.

Під час переходу, кожен комітет має B вузлів які не опрацьовують запити. Якщо $B > f$, то шард не може працювати, бо решта вузлів не можуть сформувати кворум. Звідси чим більше значення B, тим більший ризик втрати життєздатність з періодом переходу епохи.

3.5 Використання репутації для вирішення проблеми

Для покращення розподілу вузлів на шарди також можна використовувати репутацію.

Переваги використання репутації:

Баланс: кожна група буде мати загальний показник репутації, тобто будуть однакові пропорції активних, та не активних вузлів, та чесних та зловмисних валідаторів.

Стимул: для валідаторів, чим вища репутація, тим вища ймовірність бути обраним лідером групи.

В теорії сукупні показники репутація повинні запобігати зловмисній поведінці вузлів, так як вона буде не вигідна для зловмисника, а однакові пропорції активних та не активних вузлів будуть гарантувати високу швидкість транзакцій у кожній групі.

Висновки до розділу 3

У третьому розділі дипломної роботи були запропоновані найбільш ефективні методи для вирішення проблеми масштабованості блокчейну, а саме вирішення проблеми з допомогою шардингу, та оптимізація шардингу, шляхом використання захищеного середовища, для зменшення кількості груп, та використання протоколу AHL+ для оптимізації обміну повідомленнями між групами.

Також було проаналізовано варіанти і спроби реалізацій цих методів для вирішення проблеми масштабованості протоколів консенсенсу.

Обговорено деталі результатів вимірів реалізованих протоколів з використанням даних методів. А також проаналізований рівень безпеки використання даних методів у BFT протоколах консенсенсу.

ВИСНОВКИ

Результатом даної роботи є проведений аналіз опублікованих джерел за тематикою: проблеми масштабування блокчейну та методи їх вирішення. Результатом аналізу опублікованих джерел було встановлення причини проблеми масштабованості для протоколів консенсусу, що базуються на візантійській угоді, а саме в тому, що існує надлишковий обмін повідомленнями між вузлами $O(n^2)$ навіть за сприятливих мережевих умов.

Також розглянуто протоколи, які мають вирішення проблеми масштабованості алгоритмів консенсусу що базуються на візантійській угоді. В результаті прийнято рішення використовувати розбиття вузлів блокчейну на комітети, так званий метод шардингу для збільшення масштабованості блокчейну і збільшення кількості транзакцій.

В результаті роботи запропоновано використовувати такі модифікації:

1. Метод розподілу вузлів на групи – шардинг.

2. Довірене середовище виконання для зменшення кількості вузлів n потрібних для подолання візантійських відмов з $3f + 1$ до $2f + 1$, де f – скомпрометовані вузли.

3. Використовувати протокол AHL+. Для оптимізації обміну повідомленнями.

Аналіз такої комбінації показав хороші результати, при масштабуванні протоколів консенсусу блокчейну, та забезпечені безпеки при обміні транзакціями.

В наступних дослідженнях по даній тематиці потрібно буде дослідити можливості оптимального вибору кількості груп та учасників в них, для збільшення швидкості транзакцій та зменшені відмов, так як на даний момент не існує оптимальних алгоритмів вирішення даної проблеми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Basic concepts and taxonomy of dependable and secure computing. / A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr

2. Introduction to reliable and secure distributed programming / Christian Cachin, Rachid Guerraoui, and Luis Rodrigues.

3. Total order broadcast and multicast algorithms: Taxonomy and survey. / Xavier Defago, Andre Schiper, and Peter Urban. [Електронний ресурс] Режим доступу до ресурсу:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.6701&rep=rep1&type=pdf

4. The Byzantine Generals Problem / L. Lamport, R. Shostak, M. Pease [Електронний ресурс] Режим доступу до ресурсу: https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf]

5. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends [Електронний ресурс] / Zibin Zheng, Shaoan Xie // IEEE Xplire. – 2018. – Режим доступу до ресурсу:

https://ieeexplore.ieee.org/abstract/document/8029379.

6. State Machine Replication for the Masses with BFT-SMART [Електронний ресурс] / Alysson Bessani, João Sousa // IEEE Xplire. – 2014. – Режим доступу до ресурсу:

https://ieeexplore.ieee.org/abstract/document/6903593.

7. Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric) [Електронний ресурс] / Harish Sukhwan, José M. Martínez // IEEE Xplire. – 2017. – Режим доступу до ресурсу:

https://ieeexplore.ieee.org/abstract/document/8069090.

8. Bitcoin, Blockchain, and Distributed Ledgers: Between Hype and Reality [Електронний ресурс] / Ferdinando M. Ametrano // Tomorrow´s Research Today. – 2018. – Режим доступу до ресурсу:

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2832249.

9. A Secure Sharding Protocol For Open Blockchains [Електронний ресурс] / Loi Luu, Kunal Baweja // ACM Digital Library. – 2016. – Режим доступу до ресурсу:

https://dl.acm.org/citation.cfm?id=2978389.

10. RapidChain: Scaling Blockchain via Full Sharding [Електронний ресурс] // ACM Digital Library. – 2018. – Режим доступу до ресурсу: https://dl.acm.org/citation.cfm?id=3243853.

11. Blockchain Consensus Protocols in the Wild [Електронний ресурс] / Christian Cachin // Christian Cachin. – 2017. – Режим доступу до ресурсу:

https://arxiv.org/abs/1707.01873.

12. Capacity of Byzantine Consensus with Capacity-Limited Point-to-Point Links [Електронний ресурс] / Guanfeng Liang // Cornell University. – 2013. – Режим доступу до ресурсу:

https://arxiv.org/abs/1104.0043.

13. SCP: A Computationally-Scalable Byzantine Consensus Protocol For Blockchains [Електронний ресурс] / Viswesh Narayanan, Chaodong Zheng // National University of Singapore. – 2018. – Режим доступу до ресурсу:

https://www.weusecoins.com/assets/pdf/library/SCP%20-%20%20A%20Computationally-Scalable%20Byzantine.pdf.

14. Practical byzantine fault tolerance / Castro, M., Liskov, B.

[Електронний ресурс] Режим доступу до ресурсу:

http://pmg.csail.mit.edu/papers/osdi99.pdf

15. A framework for analyzing private blockchains / Dinh, T. T. A., Wang, J. [Текст]

16. Byzantine quorum systems. / Malkhi, D., and Reiter, M. [Текст]

17. Enhancing bitcoin security and performance with strong consistency via collective signing / Kogias, E. K., Jovanovic, P. [Електронний ресурс] Режим доступу до ресурсу:

https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_kokoris-kogias.pdf

18. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing [Електронний ресурс] / Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly // Usenix. – 2016. – Режим доступу до ресурсу:

https://www.usenix.org/conference/usenixsecurity16/technicalsessions/presentation/kogias.

19. A Blockchain Consensus Protocol With Horizontal Scalability / Kelong Cong [Текст]

20. The honey badger of bft protocols / A. Miller, Y. Xia, K. Croman [Електронний ресурс] Режим доступу до ресурсу:

https://eprint.iacr.org/2016/199.pdf

21. Proteus: A Scalable BFT Consensus Protocol for Blockchains / Mohammad M. Jalalzai, Costas Busch. [Електронний ресурс] Режим доступу до ресурсу: https://arxiv.org/pdf/1903.04134.pdf

22. Intel. Intel Software Guard Extensions [Електронний ресурс] / Intel – Режим доступу до ресурсу: https://software.intel.com/en-us/sgx.

23. Arm Developer. TrustZone [Електронний ресурс] / Arm Developer – Режим доступу до ресурсу: https://developer.arm.com/ip-products/security-ip/trustzone.

24. Victor Costan. Sanctum: Minimal Hardware Extensions for Strong Software Isolation [Електронний ресурс] / Victor Costan // MIT CSAIL – Режим доступу до ресурсу: https://eprint.iacr.org/2015/564.pdf.

25. George Coker. Principles of remote attestation [Електронний ресурс] / George Coker, Joshua Guttman, Peter Loscocco // Springer Link. – 2011. – Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s10207-011-0124-7.

26. Intel Software Guard Extensions. [Електронний ресурс] Режим доступу до ресурсу:

https://software.intel.com/ru-ru/sgx

27. Towards Scaling Blockchain Systems via Sharding [Електронний ресурс] / Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin Ee-Chien Chang, Qian Lin, Beng Chin Ooi // National University of Singapore. – 2019. – Режим доступу до ресурсу:

https://www.comp.nus.edu.sg/~hungdang/papers/sharding.pdf.

28. Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks [Електронний ресурс] / Jin-HeeCho // Science Direct. – 2018. – Режим доступу до ресурсу:

https://www.sciencedirect.com/science/article/pii/S1084804511000701.

29. Christian Cachin. Architecture of the Hyperledger Blockchain Fabric [Електронний ресурс] / Christian Cachin // IBM Research - Zurich. – 2016. – Режим доступу до ресурсу:

https://pdfs.semanticscholar.org/f852/c5f3fe649f8a17ded391df0796677a59927f.pdf.

30. Correctness and Fairness of Tendermint-core Blockchains [Електронний ресурс] / Yackolley Amoussou-Guenou, Antonella Del Pozzo // Cornell University. – 2018. – Режим доступу до ресурсу:

https://arxiv.org/abs/1805.08429.

31. TrustChain: A Sybil-resistant scalable blockchain [Електронний ресурс] / Pim Otte, Martijn de Vos, Johan Pouwelse // Scienct Direct. – 2017. – Режим доступу до ресурсу:

https://www.sciencedirect.com/science/article/pii/S0167739X17318988.

32. A proof of stake sharding protocol for scalable blockchains [Електронний ресурс] / Yuefei Gao, Hajime Nobuhara // PROCEEDINGS OF THE ASIA-PACIFIC ADVANCED NETWORK. – 2017. – Режим доступу до ресурсу:

http://journals.sfu.ca/apan/index.php/apan/article/view/225.

33. Using Sybil Identities for Primary User Emulation and Byzantine Attacks in DSA Networks [Електронний ресурс] / Yi Tan, Kai Hong, Shamik Senguplta // IEEE Xplire. – 2011. – Режим доступу до ресурсу:

https://ieeexplore.ieee.org/abstract/document/6134059.

34. Diverse Replication for Single-Machine Byzantine-Fault Tolerance [Електронний ресурс] / Byung-Gon Chun, Petros Maniatis // Intel Research Berkeley. – 2018. – Режим доступу до ресурсу:

https://static.usenix.org/events/usenix08/tech/full_papers/chun/chun.pdf.

35. Dynamic Root of Trust in Trusted Computing [Електронний ресурс] / Cong Nie // Helsinki University of Technology. – 2018. – Режим доступу до ресурсу:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.570.7943&rep=rep1&type=pdf.

36. ROTE: Rollback Protection for Trusted Execution [Електронний ресурс] / Sinisa Matetic, Mansoor Ahmed, Kari Kostiainen, Aritra Dhar, David Sommer, and Arthur Gervais, ETH Zurich; Ari Juels, Cornell Tech; Srdjan Capkun // Usenix.org. – 2017. – Режим доступу до ресурсу:

https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/matetic.

37. Attested append-only memory: making adversaries stick to their word [Електронний ресурс] / Byung-Gon Chun, Petros Maniatis, Scott Shenker // ACM Digital Library. – 2007. – Режим доступу до ресурсу:

https://dl.acm.org/citation.cfm?id=1294280.